Reinforcement Learning:
A Primer, Multi-Task, Goal-Conditioned

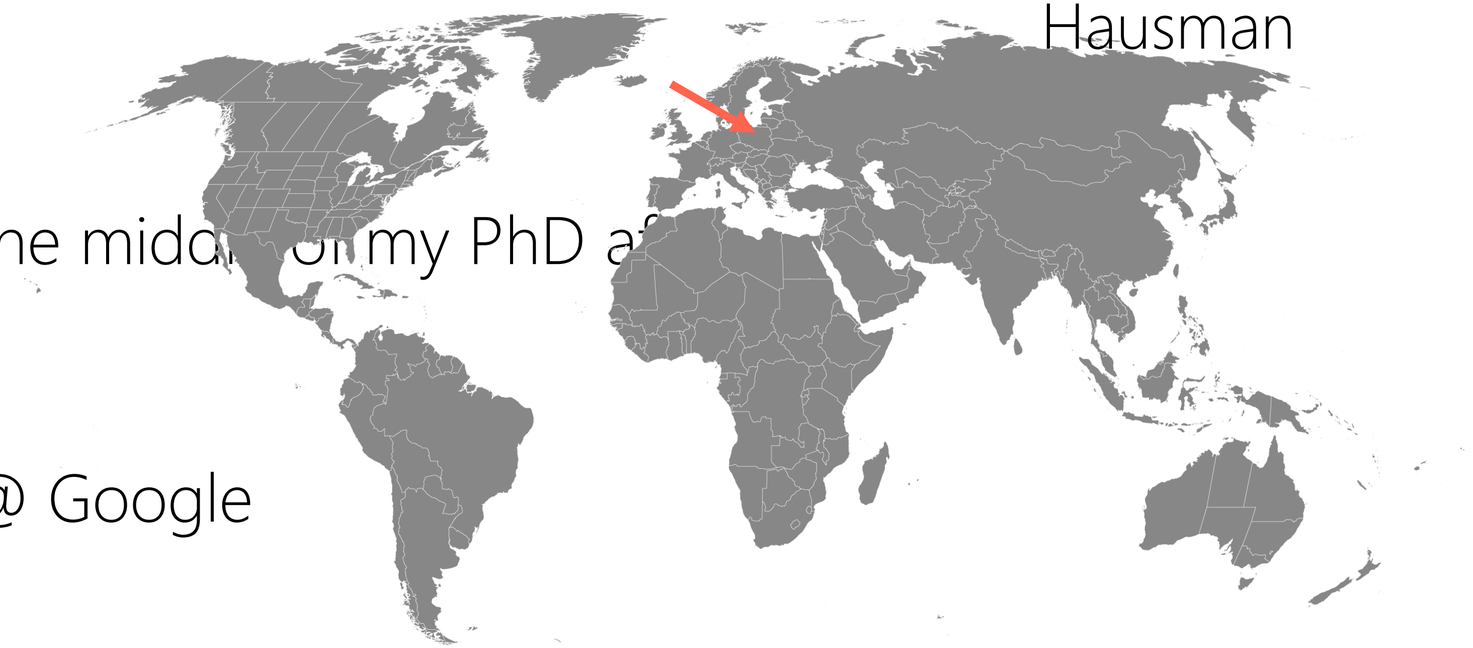
CS 330

Introduction

Some background:

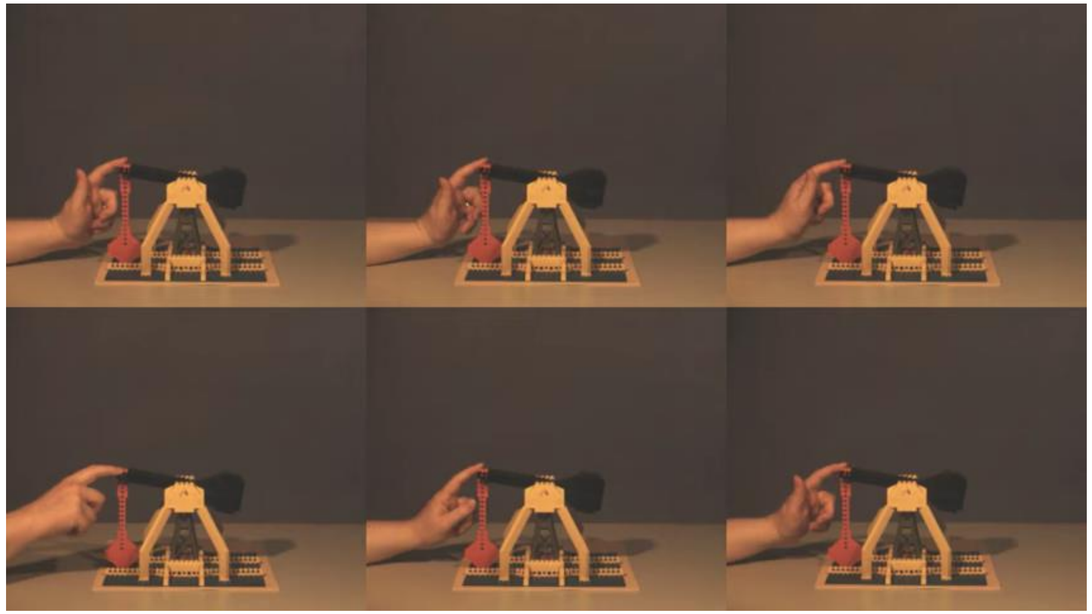
- Not a native English speaker so **please please** let me know if you don't understand something
- I like robots ☺
- Studied classical robotics first
- Got fascinated by deep RL in the middle of my PhD after a talk by Sergey Levine
- Research Scientist at Robotics @ Google

Karol Hausman



Why Reinforcement Learning?

Isolated action that doesn’t affect the future?

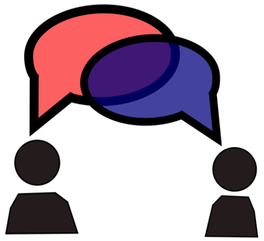


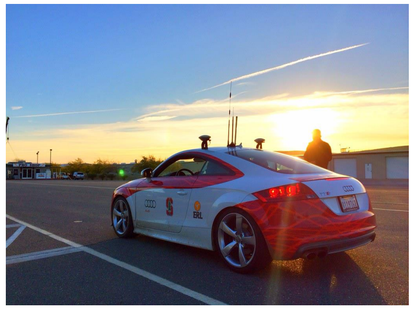
Why Reinforcement Learning?

Isolated action that doesn't affect the future?

Supervised learning?

Common applications





robotics language & dialog autonomous driving business operations finance

(most deployed ML systems)

+ a key aspect of intelligence

The Plan

Multi-task reinforcement learning problem

Policy gradients & their multi-task counterparts

Q-learning <— should be review

Multi-task Q-learning

The Plan

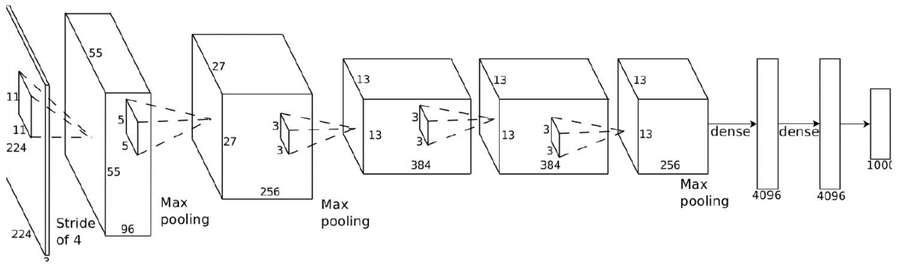
Multi-task reinforcement learning problem

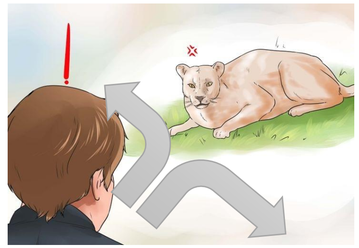
Policy gradients & their multi-task counterparts

Q-learning <— should be review

Multi-task Q-learning

Terminology & notation

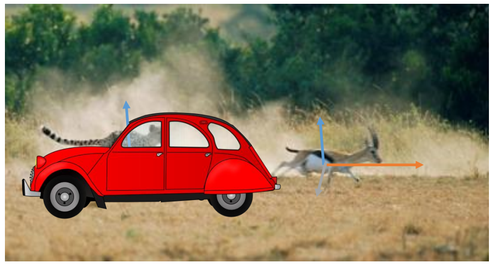




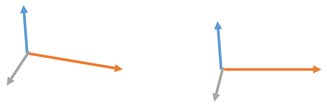
$\mathbf{o}_t$ $\pi_\theta(\mathbf{a}_t|\mathbf{o}_t)$ $\mathbf{a}_t$

$\mathbf{s}_t$ – state
$\mathbf{o}_t$ – observation
$\mathbf{a}_t$ – action

$\pi_\theta(\mathbf{a}_t|\mathbf{o}_t)$ – policy
$\pi_\theta(\mathbf{a}_t|\mathbf{s}_t)$ – policy (fully observed)



$\mathbf{o}_t$ – observation



$\mathbf{s}_t$ – state

Slide adapted from Sergey Levine

Terminology & notation

$\mathbf{o}_t$ – observation

$\mathbf{s}_t$ – state

oserved)

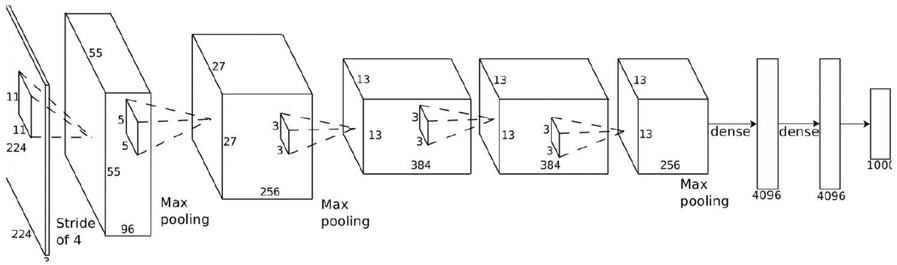
$\mathbf{s}_t$ – state

$\mathbf{o}_t$ – observation

Slide adapted from Sergey Levine

Imitation Learning

$\mathbf{o}_t$

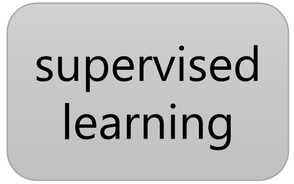


$\pi_\theta(\mathbf{a}_t|\mathbf{o}_t)$

$\mathbf{a}_t$

$\mathbf{o}_t$

$\mathbf{a}_t$

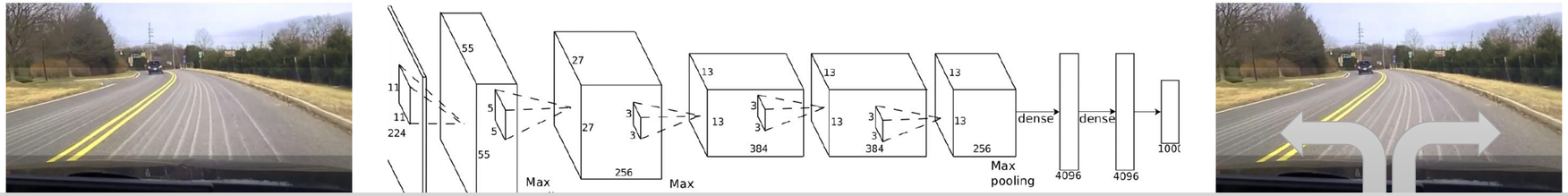


$\pi_\theta(\mathbf{a}_t|\mathbf{o}_t)$

Images: Bojarski et al. ‘16, NVIDIA

Slide adapted from Sergey Levine

Imitation Learning



Imitation Learning vs Reinforcement Learning?

$\mathbf{o}_t$

$\mathbf{a}_t$

supervised learning

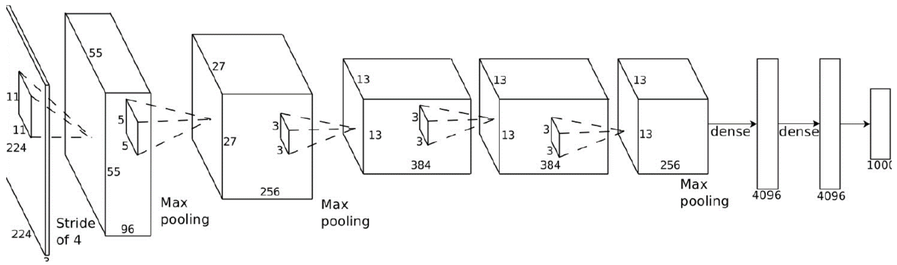
$\pi_\theta(\mathbf{a}_t|\mathbf{o}_t)$

Images: Bojarski et al. '16, NVIDIA

Slide adapted from Sergey Levine

Reward functions

$\mathbf{o}_t$



$\pi_\theta(\mathbf{a}_t|\mathbf{o}_t)$

$\mathbf{a}_t$

which action is better or worse?

$r(\mathbf{s}, \mathbf{a})$: reward function

tells us which states and actions are better

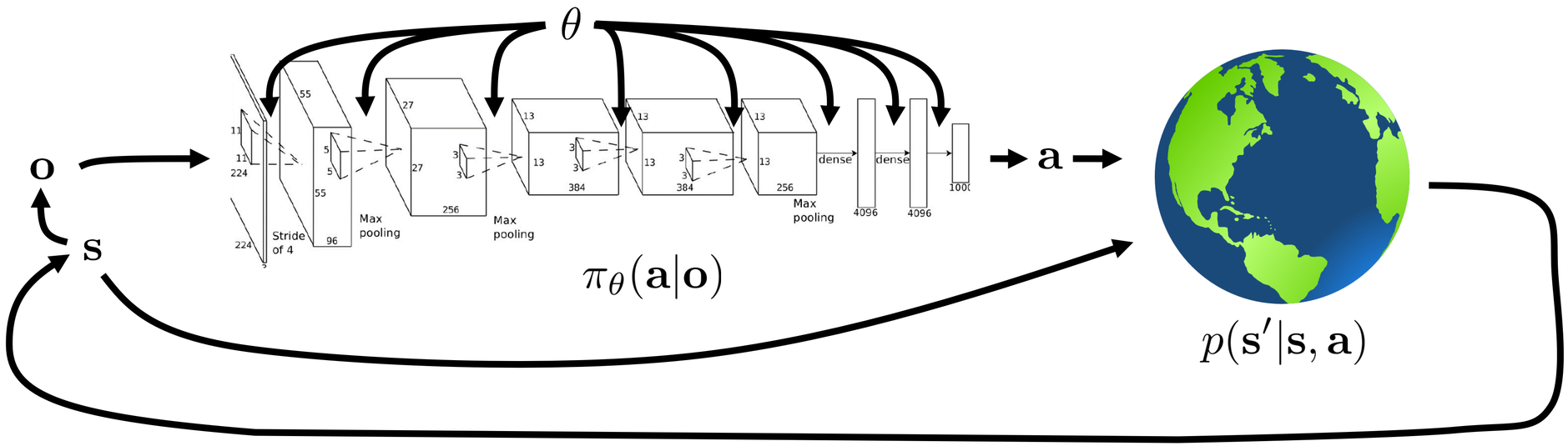
$\mathbf{s}$, $\mathbf{a}$, $r(\mathbf{s}, \mathbf{a})$, and $p(\mathbf{s}'|\mathbf{s}, \mathbf{a})$ define Markov decision process

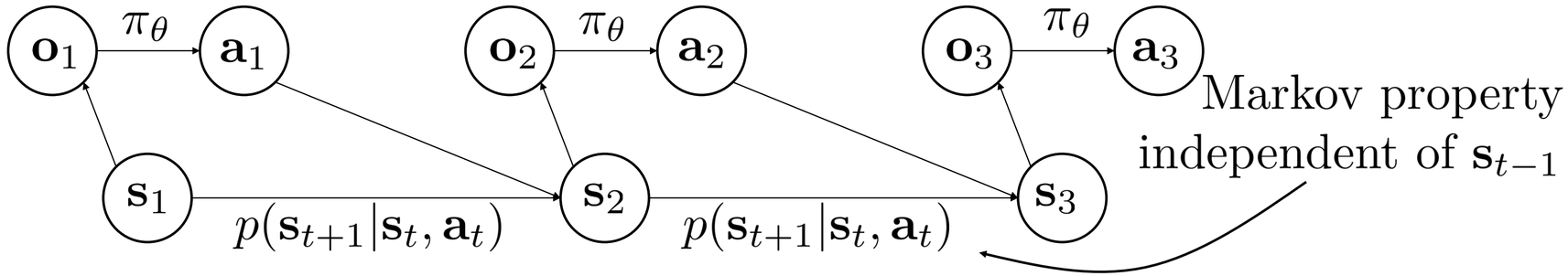
high reward

low reward

Slide adapted from Sergey Levine

The goal of reinforcement learning





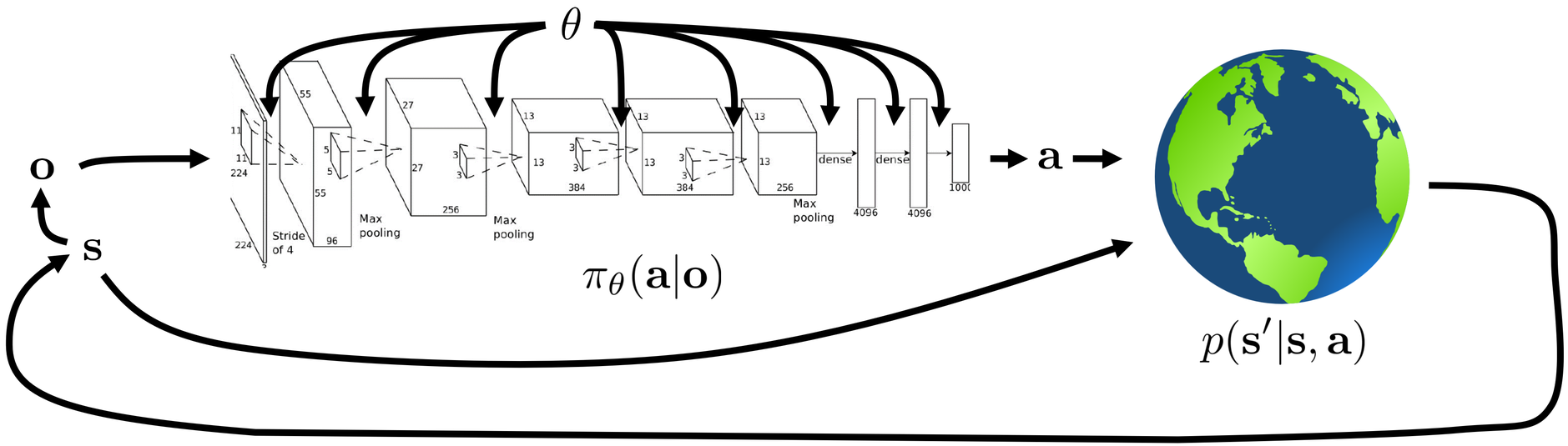
Slide adapted from Sergey Levine

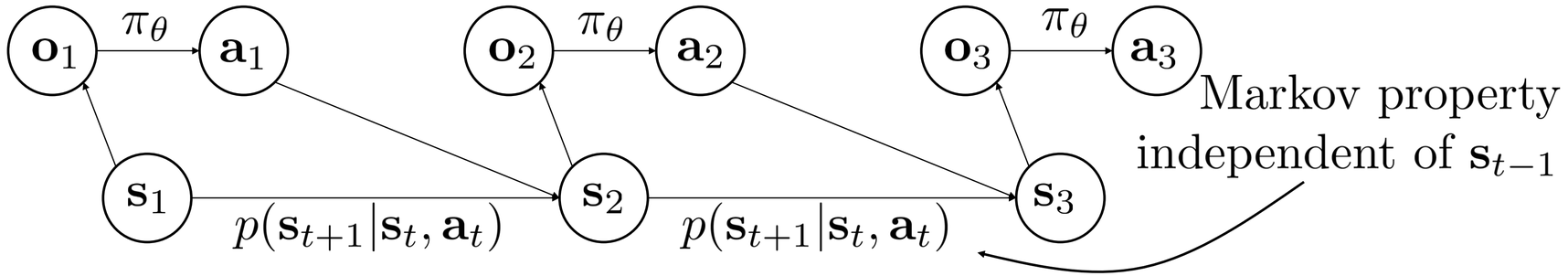
Partial observability

Fully observable?

- Simulated robot performing a reaching task given the goal position and positions and velocities of all of its joints
- Indiscriminate robotic grasping from a bin given an overhead image
- A robot sorting trash given a camera image

The goal of reinforcement learning





$$\theta^\star = \arg\max_\theta E_{(\mathbf{s},\mathbf{a})\sim p_\theta(\mathbf{s},\mathbf{a})}[r(\mathbf{s},\mathbf{a})]$$

infinite horizon case

$$\theta^\star = \arg\max_\theta \sum_{t=1}^{T} E_{(\mathbf{s}_t,\mathbf{a}_t)\sim p_\theta(\mathbf{s}_t,\mathbf{a}_t)}[r(\mathbf{s}_t,\mathbf{a}_t)]$$

finite horizon case

Slide adapted from Sergey Levine

What is a reinforcement learning task?

Recall: supervised learning

data generating distributions, loss

A task: $\mathcal{T}_i \triangleq \{p_i(\mathbf{x}), p_i(\mathbf{y}|\mathbf{x}), \mathcal{L}_i\}$

Reinforcement learning

action space dynamics

A task: $\mathcal{T}_i \triangleq \{\mathcal{S}_i, \mathcal{A}_i, p_i(\mathbf{s}_1), p_i(\mathbf{s}'|\mathbf{s}, \mathbf{a}), r_i(\mathbf{s}, \mathbf{a})\}$

state space

initial state distribution

reward

a Markov decision process

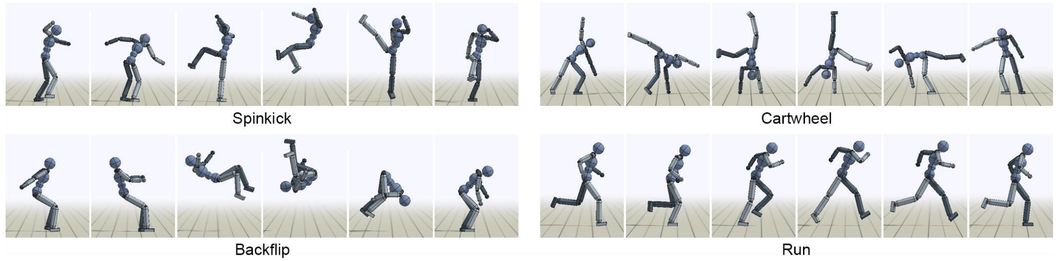
much more than the semantic meaning of task!

Examples Task Distributions

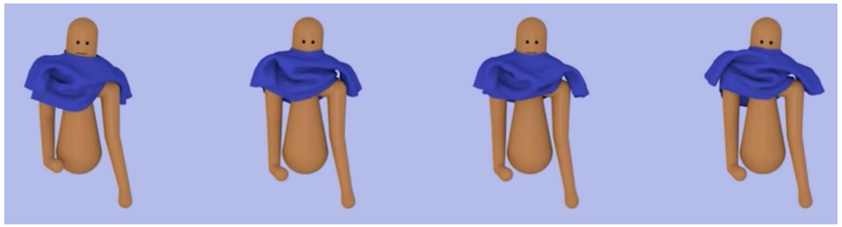
A task: $\mathcal{T}_i \triangleq \{\mathcal{S}_i, \mathcal{A}_i, p_i(\mathbf{s}_1), p_i(\mathbf{s}'|\mathbf{s},\mathbf{a}), r_i(\mathbf{s},\mathbf{a})\}$

Character animation:

across maneuvers
$r_i(\mathbf{s},\mathbf{a})$ vary



across garments &
initial states
$p_i(\mathbf{s}_1), p_i(\mathbf{s}'|\mathbf{s},\mathbf{a})$ vary



Multi-robot RL:

$\mathcal{S}_i, \mathcal{A}_i, p_i(\mathbf{s}_1), p_i(\mathbf{s}'|\mathbf{s},\mathbf{a})$ vary

What is a reinforcement learning task?

Reinforcement learning

A task: $\mathcal{T}_i \triangleq \{\mathcal{S}_i, \mathcal{A}_i, p_i(\mathbf{s}_1), p_i(\mathbf{s}'|\mathbf{s},\mathbf{a}), r_i(\mathbf{s},\mathbf{a})\}$

action space — dynamics

state space — initial state distribution — reward

An alternative view:

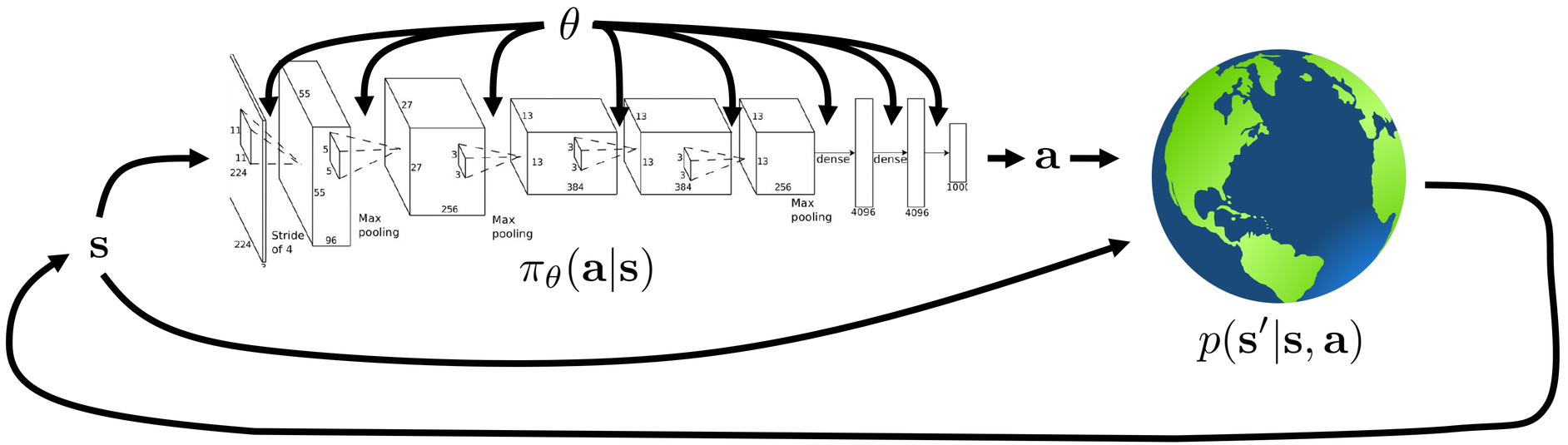
A **task identifier** is part of the state: $\mathbf{s} = (\overline{\mathbf{s}}, \mathbf{z}_i)$

original state

$\mathcal{T}_i \triangleq \{\mathcal{S}_i, \mathcal{A}_i, p_i(\mathbf{s}_1), p(\mathbf{s}'|\mathbf{s},\mathbf{a}), r(\mathbf{s},\mathbf{a})\} \longrightarrow \{\mathcal{T}_i\} = \left\{\cup\mathcal{S}_i, \cup\mathcal{A}_i, \frac{1}{N}\sum_i p_i(\mathbf{s}_1), p(\mathbf{s}'|\mathbf{s},\mathbf{a}), r(\mathbf{s},\mathbf{a})\right\}$

It can be cast as a **standard Markov decision process**!

The goal of multi-task reinforcement learning



Multi-task RL

The same as before, **except**:

a **task identifier** is part of the state: $\mathbf{s} = (\overline{\mathbf{s}}, \mathbf{z}_i)$

e.g. one-hot task ID

language description

desired goal state, $\mathbf{z}_i = \mathbf{s}_g$ ⟵ "**goal-conditioned** RL"

If it's still a **standard Markov decision process**,

then, why not apply standard RL algorithms? You can! You can often do better.

What is the reward?

The same as before

Or, for goal-conditioned RL:

$r(\mathbf{s}) = r(\overline{\mathbf{s}}, \mathbf{s}_g) = -d(\overline{\mathbf{s}}, \mathbf{s}_g)$

Distance function d examples:

- Euclidean ℓ_2
- sparse 0/1

The Plan

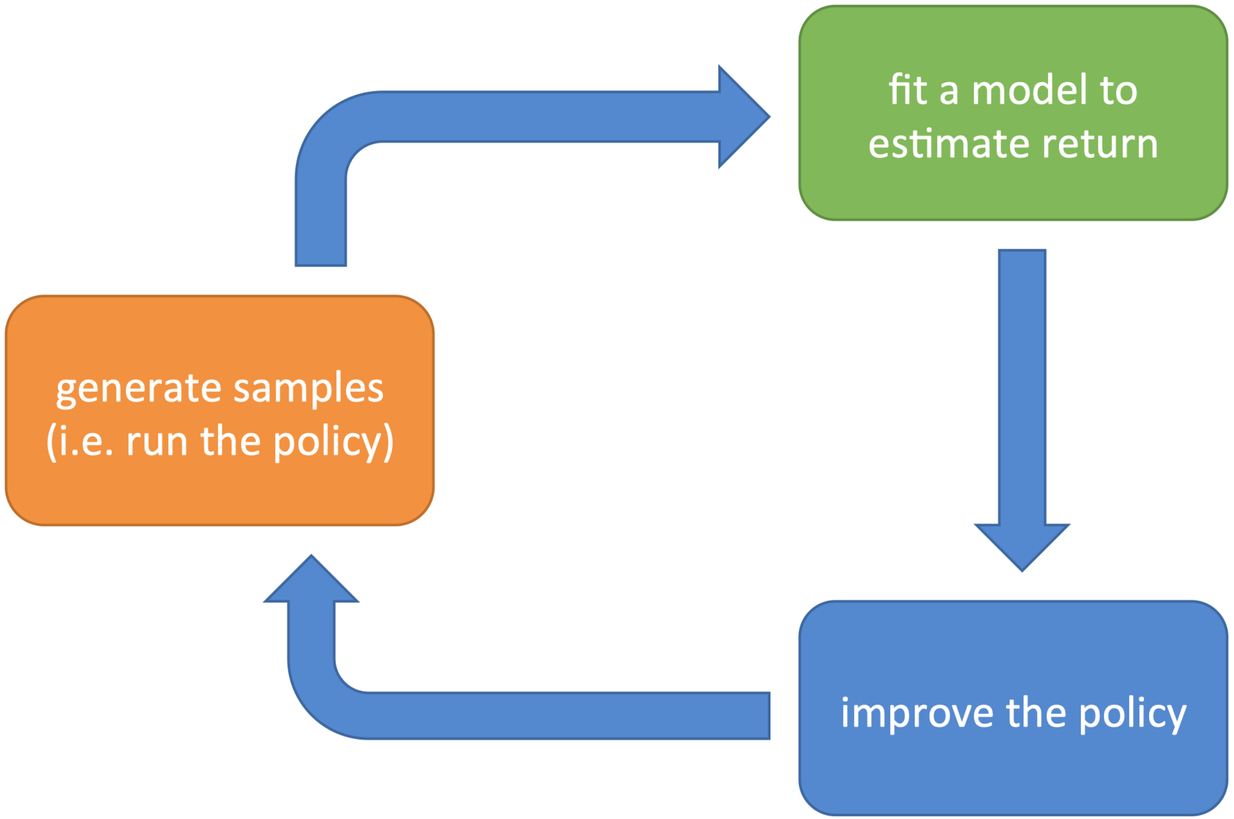
Multi-task reinforcement learning problem

Policy gradients & their multi-task counterparts

Q-learning

Multi-task Q-learning

The anatomy of a reinforcement learning algorithm



compute $\hat{Q} = \sum_{t'=t}^{T} \gamma^{t'-t} r_{t'}$ (MC policy gradient)
fit $Q_\phi(\mathbf{s}, \mathbf{a})$ (actor-critic, Q-learning)
estimate $p(\mathbf{s}'|\mathbf{s}, \mathbf{a})$ (model-based)

$\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$ (policy gradient)
$\pi(\mathbf{s}) = \arg\max Q_\phi(\mathbf{s}, \mathbf{a})$ (Q-learning)
optimize $\pi_\theta(\mathbf{a}|\mathbf{s})$ (model-based)

This lecture: focus on model-free RL methods (policy gradient, Q-learning)

10/19: focus on model-based RL methods

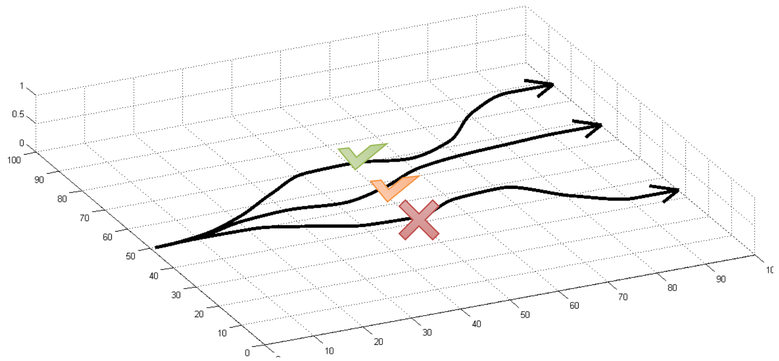
On-policy vs Off-policy

- Data comes from the current policy
- Compatible with all RL algorithms
- Can’t reuse data from previous policies

- Data comes from any policy
- Works with specific RL algorithms
- Much more sample efficient, can re-use old data

Evaluating the objective

$$\theta^\star = \arg\max_\theta \underbrace{E_{\tau\sim p_\theta(\tau)}\left[\sum_t r(\mathbf{s}_t, \mathbf{a}_t)\right]}_{J(\theta)}$$



$$J(\theta) = E_{\tau\sim p_\theta(\tau)}\left[\sum_t r(\mathbf{s}_t, \mathbf{a}_t)\right] \approx \frac{1}{N}\sum_i\sum_t r(\mathbf{s}_{i,t}, \mathbf{a}_{i,t})$$

sum over samples from π_θ

Slide adapted from Sergey Levine

Direct policy differentiation

$$\theta^\star = \arg\max_\theta \underbrace{E_{\tau\sim p_\theta(\tau)}\left[\sum_t r(\mathbf{s}_t, \mathbf{a}_t)\right]}_{J(\theta)}$$

a convenient identity

$$\pi_\theta(\tau)\nabla_\theta \log \pi_\theta(\tau) = \pi_\theta(\tau)\frac{\nabla_\theta \pi_\theta(\tau)}{\pi_\theta(\tau)} = \nabla_\theta \pi_\theta(\tau)$$

$$J(\theta) = E_{\tau\sim\pi_\theta(\tau)}[\underbrace{r(\tau)}_{\sum_{t=1}^{T} r(\mathbf{s}_t, \mathbf{a}_t)}] = \int \pi_\theta(\tau) r(\tau) d\tau$$

$$\nabla_\theta J(\theta) = \int \nabla_\theta \pi_\theta(\tau) r(\tau) d\tau = \int \pi_\theta(\tau)\nabla_\theta \log \pi_\theta(\tau) r(\tau) d\tau = E_{\tau\sim\pi_\theta(\tau)}[\nabla_\theta \log \pi_\theta(\tau) r(\tau)]$$

Slide adapted from Sergey Levine

Direct policy differentiation

$$\theta^\star = \arg\max_\theta J(\theta)$$

$$J(\theta) = E_{\tau\sim\pi_\theta(\tau)}[r(\tau)]$$

$$\nabla_\theta J(\theta) = E_{\tau\sim\pi_\theta(\tau)}[\nabla_\theta \log \pi_\theta(\tau) r(\tau)]$$

$$\underbrace{\pi_\theta(\mathbf{s}_1, \mathbf{a}_1, \ldots, \mathbf{s}_T, \mathbf{a}_T)}_{\pi_\theta(\tau)} = p(\mathbf{s}_1)\prod_{t=1}^{T}\pi_\theta(\mathbf{a}_t|\mathbf{s}_t)p(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)$$

log of both sides

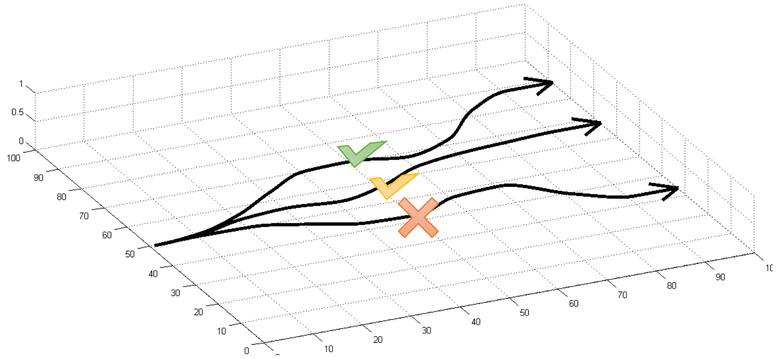
$$\log \pi_\theta(\tau) = \log p(\mathbf{s}_1) + \sum_{t=1}^{T}\log \pi_\theta(\mathbf{a}_t|\mathbf{s}_t) + \log p(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)$$

$$\nabla_\theta \left[\log p(\mathbf{s}_1) + \sum_{t=1}^{T}\log \pi_\theta(\mathbf{a}_t|\mathbf{s}_t) + \log p(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)\right]$$

$$\nabla_\theta J(\theta) = E_{\tau\sim\pi_\theta(\tau)}\left[\left(\sum_{t=1}^{T}\nabla_\theta \log \pi_\theta(\mathbf{a}_t|\mathbf{s}_t)\right)\left(\sum_{t=1}^{T} r(\mathbf{s}_t, \mathbf{a}_t)\right)\right]$$

Slide adapted from Sergey Levine

Evaluating the policy gradient



$$\text{recall: } J(\theta) = E_{\tau \sim p_\theta(\tau)} \left[\sum_t r(\mathbf{s}_t, \mathbf{a}_t) \right] \approx \frac{1}{N} \sum_i \sum_t r(\mathbf{s}_{i,t}, \mathbf{a}_{i,t})$$

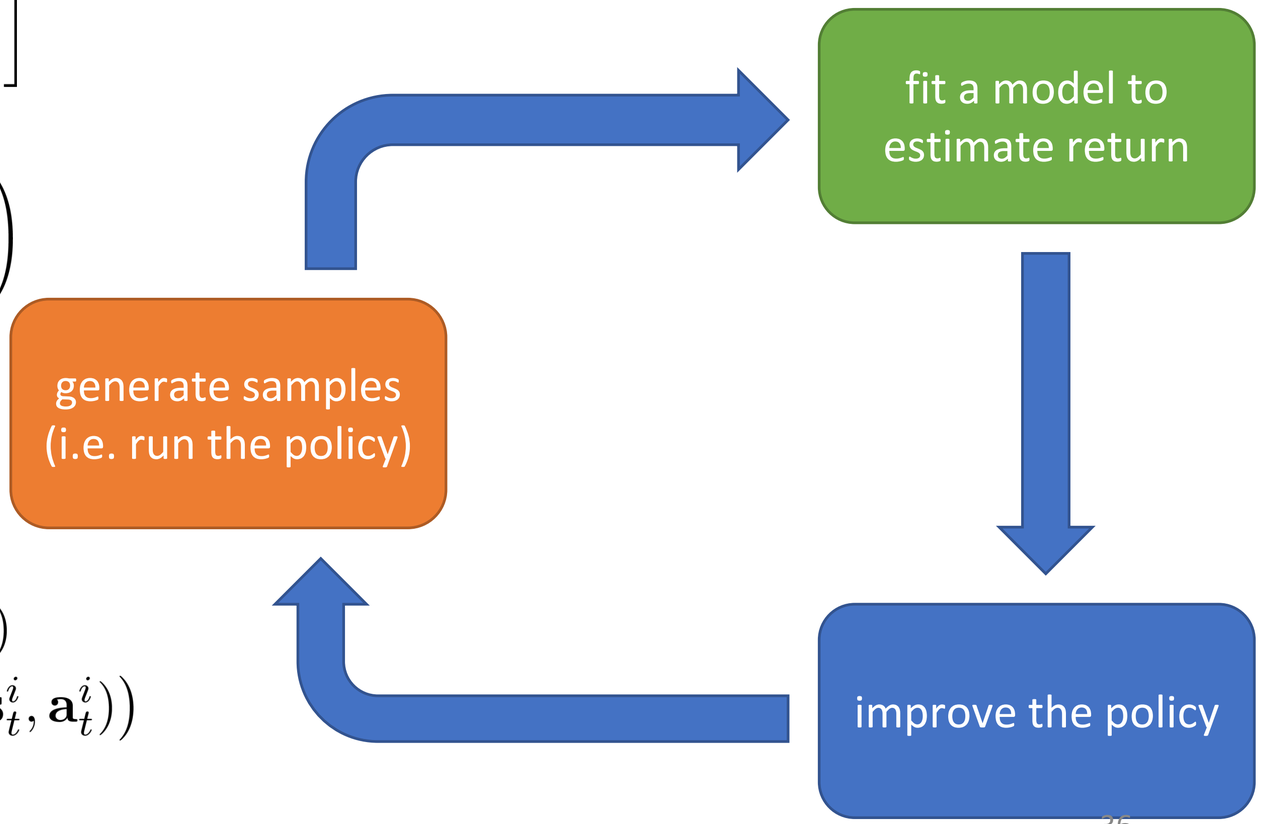
$$\nabla_\theta J(\theta) = E_{\tau \sim \pi_\theta(\tau)} \left[\left(\sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(\mathbf{a}_t | \mathbf{s}_t) \right) \left(\sum_{t=1}^{T} r(\mathbf{s}_t, \mathbf{a}_t) \right) \right]$$

$$\nabla_\theta J(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \left(\sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(\mathbf{a}_{i,t} | \mathbf{s}_{i,t}) \right) \left(\sum_{t=1}^{T} r(\mathbf{s}_{i,t}, \mathbf{a}_{i,t}) \right)$$

$$\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$$

REINFORCE algorithm:

1. sample $\{\tau^i\}$ from $\pi_\theta(\mathbf{a}_t | \mathbf{s}_t)$ (run the policy)
2. $\nabla_\theta J(\theta) \approx \sum_i \left(\sum_t \nabla_\theta \log \pi_\theta(\mathbf{a}_t^i | \mathbf{s}_t^i) \right) \left(\sum_t r(\mathbf{s}_t^i, \mathbf{a}_t^i) \right)$
3. $\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$



Slide adapted from Sergey Levine

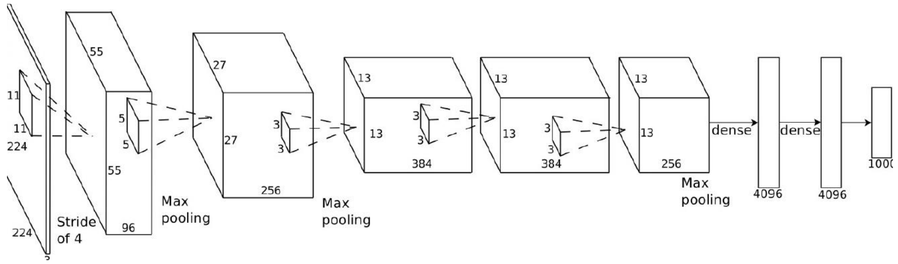
Comparison to maximum likelihood

$$\text{policy gradient:} \quad \nabla_\theta J(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \left(\sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(\mathbf{a}_{i,t}|\mathbf{s}_{i,t}) \right) \left(\sum_{t=1}^{T} r(\mathbf{s}_{i,t}, \mathbf{a}_{i,t}) \right)$$

Multi-task learning algorithms can readily be applied!

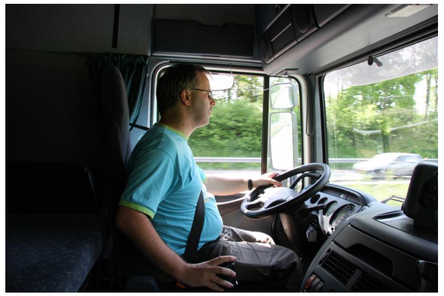
$$\text{maximum likelihood:} \quad \nabla_\theta J_{\text{ML}}(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \left(\sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(\mathbf{a}_{i,t}|\mathbf{s}_{i,t}) \right)$$

$\mathbf{s}_t$



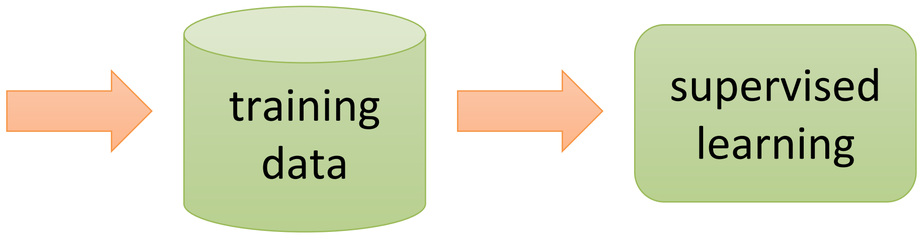
$\pi_\theta(\mathbf{a}_t|\mathbf{s}_t)$

$\mathbf{a}_t$



$\mathbf{s}_t$

$\mathbf{a}_t$



$\pi_\theta(\mathbf{a}_t|\mathbf{s}_t)$

Slide adapted from Sergey Levine

What did we just do?

$$\nabla_\theta J(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \left(\sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(\mathbf{a}_{i,t}|\mathbf{s}_{i,t}) \right) \left(\sum_{t=1}^{T} r(\mathbf{s}_{i,t}, \mathbf{a}_{i,t}) \right)$$

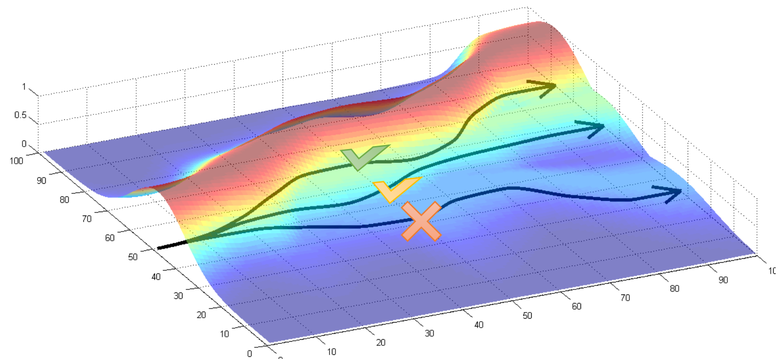
$$\nabla_\theta J(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \underbrace{\nabla_\theta \log \pi_\theta(\tau_i) r(\tau_i)}_{\sum_{t=1}^{T} \nabla_\theta \log_\theta \pi_\theta(\mathbf{a}_{i,t}|\mathbf{s}_{i,t})}$$

maximum likelihood: $\nabla_\theta J_{\mathrm{ML}}(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \nabla_\theta \log \pi_\theta(\tau_i)$

good stuff is made more likely

bad stuff is made less likely

simply formalizes the notion of “trial and error”!



REINFORCE algorithm:

1. sample $\{\tau^i\}$ from $\pi_\theta(\mathbf{a}_t|\mathbf{s}_t)$ (run it on the robot)
2. $\nabla_\theta J(\theta) \approx \sum_i \left(\sum_t \nabla_\theta \log \pi_\theta(\mathbf{a}_t^i|\mathbf{s}_t^i) \right) \left(\sum_t r(\mathbf{s}_t^i, \mathbf{a}_t^i) \right)$
3. $\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$

Slide adapted from Sergey Levine

Policy Gradients

$$\text{policy gradient:} \quad \nabla_\theta J(\theta) = E_{\tau \sim \pi_\theta(\tau)} \left[\left(\sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(\mathbf{a}_t | \mathbf{s}_t) \right) \left(\sum_{t=1}^{T} r(\mathbf{s}_t, \mathbf{a}_t) \right) \right]$$

Pros:

+ Simple
+ Easy to combine with **existing multi-task & meta-learning algorithms**

Cons:

- Produces a **high-variance** gradient
  - Can be mitigated with **baselines** (used by all algorithms in practice), trust regions
- Requires **on-policy** data
  - Cannot reuse existing experience to estimate the gradient!
    - **Importance weights** can help, but also high variance

The Plan

Multi-task reinforcement learning problem

Policy gradients & their multi-task/meta counterparts

Q-learning

Multi-task Q-learning

Value-Based RL: Definitions

Value function: $V^{\pi}(\mathbf{s}_t) = \sum_{t'=t}^{T} \mathbb{E}_{\pi}\left[r(\mathbf{s}_{t'}, \mathbf{a}_{t'}) \mid \mathbf{s}_t\right]$ total reward starting from $\mathbf{s}$ and following π

"how good is a state"

Q function: $Q^{\pi}(\mathbf{s}_t, \mathbf{a}_t) = \sum_{t'=t}^{T} \mathbb{E}_{\pi}\left[r(\mathbf{s}_{t'}, \mathbf{a}_{t'}) \mid \mathbf{s}_t, \mathbf{a}_t\right]$ total reward starting from $\mathbf{s}$, taking $\mathbf{a}$, and then following π

"how good is a state-action pair"

They're related: $V^{\pi}(\mathbf{s}_t) = \mathbb{E}_{\mathbf{a}_t \sim \pi(\cdot|\mathbf{s}_t)}[Q^{\pi}(\mathbf{s}_t, \mathbf{a}_t)]$

If you know Q^{π}, you can use it to improve π.

Set $\pi(\mathbf{a}|\mathbf{s}) \leftarrow 1 \; for \; \mathbf{a} = \arg\max_{\bar{\mathbf{a}}} Q^{\pi}(\mathbf{s}, \bar{\mathbf{a}})$ New policy is at least as good as old policy.

Value-Based RL: Definitions

Value function: $V^{\pi}(\mathbf{s}_t) = \sum_{t'=t}^{T} \mathbb{E}_{\pi}\left[r(\mathbf{s}_{t'}, \mathbf{a}_{t'}) \mid \mathbf{s}_t\right]$ total reward starting from $\mathbf{s}$ and following π

"how good is a state"

Q function: $Q^{\pi}(\mathbf{s}_t, \mathbf{a}_t) = \sum_{t'=t}^{T} \mathbb{E}_{\pi}\left[r(\mathbf{s}_{t'}, \mathbf{a}_{t'}) \mid \mathbf{s}_t, \mathbf{a}_t\right]$ total reward starting from $\mathbf{s}$, taking $\mathbf{a}$, and then following π

"how good is a state-action pair"

For the optimal policy $\pi^{\star}$: $Q^{\star}(\mathbf{s}_t, \mathbf{a}_t) = \mathbb{E}_{\mathbf{s}' \sim p(\cdot|\mathbf{s},\mathbf{a})}\left[r(\mathbf{s}, \mathbf{a}) + \gamma \max_{\mathbf{a}'} Q^{\star}(\mathbf{s}', \mathbf{a}')\right]$

Bellman equation

Value-Based RL

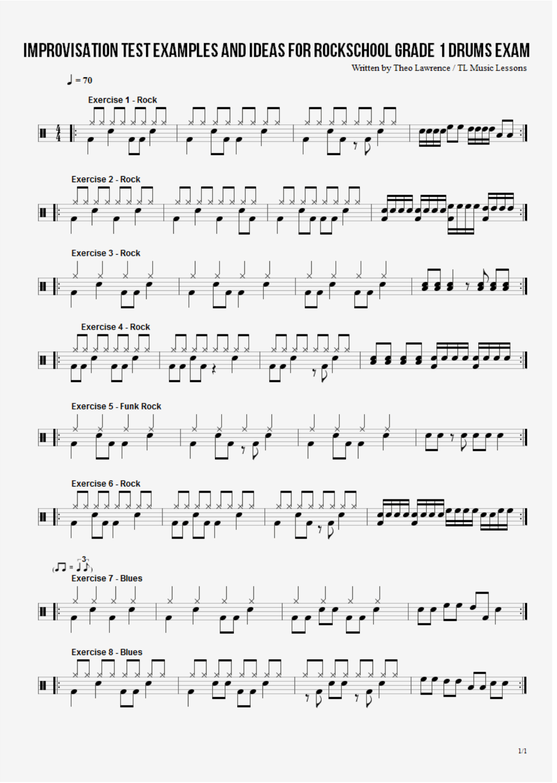
Value function: $V^{\pi}(\mathbf{s}_t) = ?$

Q function: $Q^{\pi}(\mathbf{s}_t, \mathbf{a}_t) = ?$

Q* function: $Q^{*}(\mathbf{s}_t, \mathbf{a}_t) = ?$

Value* function: $V^{*}(\mathbf{s}_t) = ?$

Reward = 1 if I can play it in a month, 0 otherwise



Set $\pi(\mathbf{a}|\mathbf{s}) \leftarrow 1$ for $\mathbf{a} = \arg\max_{\bar{\mathbf{a}}} Q^{\pi}(\mathbf{s}, \bar{\mathbf{a}})$ New policy is at least as good as old policy.

Current $\pi(\mathbf{a}_1|\mathbf{s}) = 1$

Fitted Q-iteration Algorithm

full fitted Q-iteration algorithm:

$K\times$

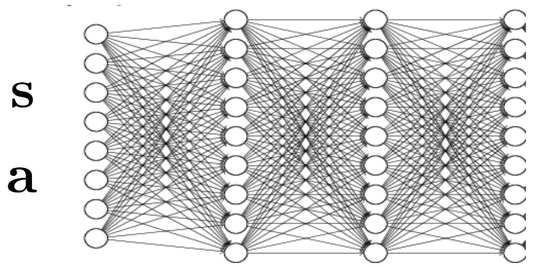
1. collect dataset $\{(\mathbf{s}_i, \mathbf{a}_i, \mathbf{s}'_i, r_i)\}$ using some policy
2. set $\mathbf{y}_i \leftarrow r(\mathbf{s}_i, \mathbf{a}_i) + \gamma \max_{\mathbf{a}'_i} Q_\phi(\mathbf{s}'_i, \mathbf{a}'_i)$
3. set $\phi \leftarrow \arg\min_\phi \frac{1}{2} \sum_i \left\| Q_\phi(\mathbf{s}_i, \mathbf{a}_i) - \mathbf{y}_i \right\|^2$

Algorithm hyperparameters

dataset size N, collection policy

iterations K

gradient steps S



$Q_\phi(\mathbf{s}, \mathbf{a})$
parameters ϕ

Result: get a policy $\boldsymbol{\pi(\mathbf{a}|\mathbf{s})}$ from $\boldsymbol{\arg\max_{\mathbf{a}} Q_\phi(\mathbf{s}, \mathbf{a})}$

Important notes:

We can **reuse data** from previous policies!
an **off-policy** algorithm using replay buffers

This is **not** a **gradient descent** algorithm!

Can be readily extended to multi-task/goal-conditioned RL

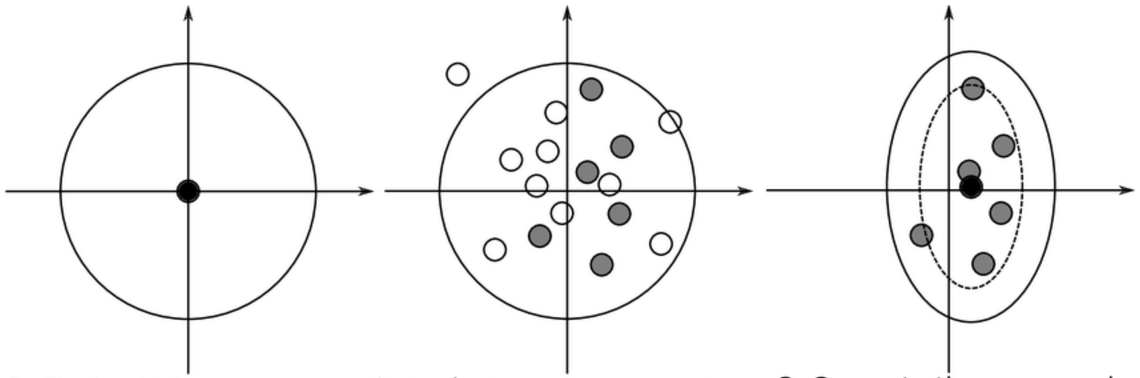
Slide adapted from Sergey Levine

Example: Q-learning Applied to Robotics

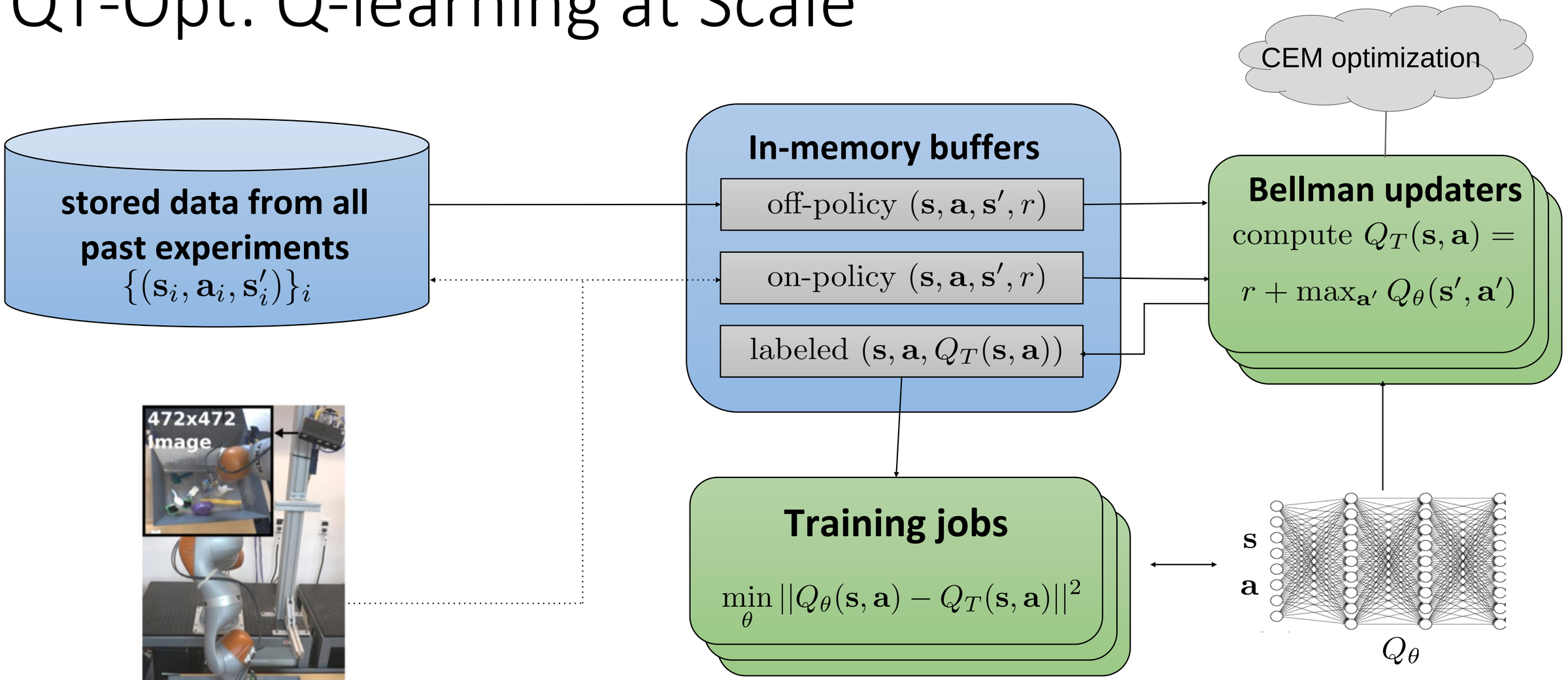
1. collect dataset $\{(\mathbf{s}_i, \mathbf{a}_i, \mathbf{s}'_i, r_i)\}$ using some policy
2. set $\mathbf{y}_i \leftarrow r(\mathbf{s}_i, \mathbf{a}_i) + \gamma \max_{\mathbf{a}'_i} Q_\phi(\mathbf{s}'_i, \mathbf{a}'_i)$
3. set $\phi \leftarrow \arg\min_\phi \frac{1}{2} \sum_i \|Q_\phi(\mathbf{s}_i, \mathbf{a}_i) - \mathbf{y}_i\|^2$

Continuous action space?

Simple optimization algorithm -> Cross Entropy Method (CEM)



QT-Opt: Q-learning at Scale

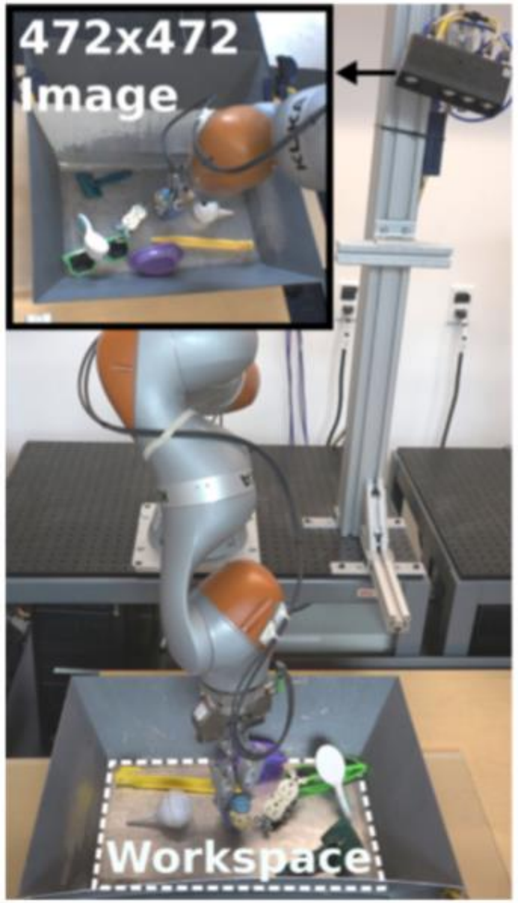


$$\text{minimize} \sum_i (Q(\mathbf{s}_i, \mathbf{a}_i) - [r(\mathbf{s}_i, \mathbf{a}_i) + \max_{\mathbf{a}'_i} Q(\mathbf{s}'_i, \mathbf{a}'_i)])^2$$

Slide adapted from D. Kalashnikov

QT-Opt: Kalashnikov et al. '18, Google Brain

QT-Opt: MDP Definition for Grasping

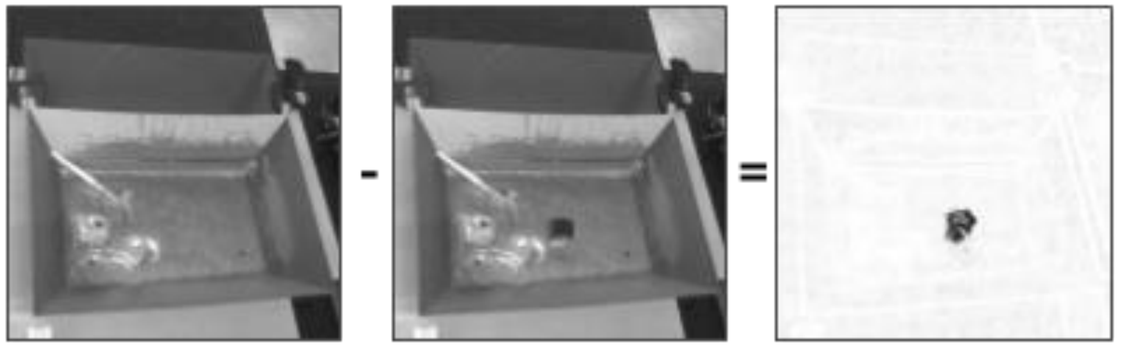


State: over the shoulder RGB camera image, no depth

Action: 4DOF pose change in Cartesian space + gripper control

Reward: binary reward at the end, if the object was lifted. Sparse. No shaping

Automatic success detection:



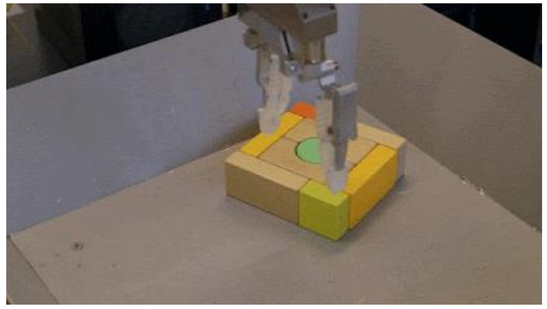
Slide adapted from D. Kalashnikov

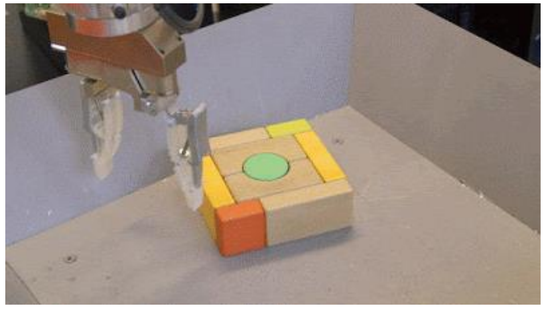
QT-Opt: Setup and Results

7 robots collected 580k grasps

Unseen test objects

96% test success rate!





Q-learning

Bellman equation: $Q^\star(\mathbf{s}_t, \mathbf{a}_t) = \mathbb{E}_{\mathbf{s}' \sim p(\cdot|\mathbf{s},\mathbf{a})}\left[r(\mathbf{s},\mathbf{a}) + \gamma \max_{\mathbf{a}'} Q^\star(\mathbf{s}', \mathbf{a}')\right]$

Pros:

+ More **sample efficient** than on-policy methods
+ Can incorporate **off-policy data** (including a fully offline setting)
+ Can updates the policy even without seeing the reward
+ Relatively easy to parallelize

Cons:

- Harder to apply standard meta-learning algorithms (DP algorithm)
- Lots of “tricks” to make it work
- Potentially could be harder to learn than just a policy

The Plan

Multi-task reinforcement learning problem

Policy gradients & their multi-task/meta counterparts

Q-learning

Multi-task Q-learning

Multi-Task RL Algorithms

Policy: $\pi_\theta(\mathbf{a}|\overline{\mathbf{s}}) \;\; \rightarrow \;\; \pi_\theta(\mathbf{a}|\overline{\mathbf{s}}, \mathbf{z}_i)$

Q-function: $Q_\phi(\overline{\mathbf{s}}, \mathbf{a}) \;\; \rightarrow \;\; Q_\phi(\overline{\mathbf{s}}, \mathbf{a}, \mathbf{z}_i)$

Analogous to multi-task supervised learning: stratified sampling, soft/hard weight sharing, etc.

What is different about **reinforcement learning**?

The **data distribution** is controlled by the agent!

Why mention it now?

Should we share **data** in addition to sharing **weights**?

An example

Task 1: passing

Task 2: shooting goals

What if you accidentally perform a good pass when trying to shoot a goal?

Store experience as normal. *and* **Relabel** experience with task 2 ID & reward and store.

“hindsight relabeling” "hindsight experience replay” (HER)

Goal-conditioned RL with hindsight relabeling

$k++$

1. Collect data $\mathcal{D}_k = \{(\mathbf{s}_{1:T}, \mathbf{a}_{1:T}, \mathbf{s}_g, r_{1:T})\}$ using some policy
2. Store data in replay buffer $\mathcal{D} \leftarrow \mathcal{D} \cup \mathcal{D}_k$
3. Perform hindsight relabeling:
   a. Relabel experience in $\mathcal{D}_k$ using last state as goal:
      $\mathcal{D}'_k = \{(\mathbf{s}_{1:T}, \mathbf{a}_{1:T}, \mathbf{s}_T, r'_{1:T}\}$ where $r'_t = -d(\mathbf{s}_t, \mathbf{s}_T)$
   b. Store relabeled data in replay buffer $\mathcal{D} \leftarrow \mathcal{D} \cup \mathcal{D}'_k$
4. Update policy using replay buffer $\mathcal{D}$

< strategies?

e trajectory

Result: exploration challenges alleviated

Kaelbling. Learning to Achieve Goals. IJCAI '93
Andrychowicz et al. Hindsight Experience Replay. NeurIPS '17

Why mention it now?

Task 1: close a drawer

Task 2: open a drawer

Can we use episodes from drawer opening task for drawer closing task?

How does that answer change for Q-learning vs Policy Gradient?

Multi-task RL with relabeling

1. Collect data $\mathcal{D}_k = \{(\mathbf{s}_{1:T}, \mathbf{a}_{1:T}, \mathbf{z}_i, r_{1:T})\}$ using some policy
2. Store data in replay buffer $\mathcal{D} \leftarrow \mathcal{D} \cup \mathcal{D}_k$
3. Perform hindsight relabeling:
   a. Relabel experience in $\mathcal{D}_k$ for task $\mathcal{T}_j$:
      $\mathcal{D}'_k = \{(\mathbf{s}_{1:T}, \mathbf{a}_{1:T}, \mathbf{z}_j, r'_{1:T}\}$ where $r'_t = r_j(\mathbf{s}_t)$
   b. Store relabeled data in replay buffer $\mathcal{D} \leftarrow \mathcal{D} \cup \mathcal{D}'_k$
4. Update policy using replay buffer $\mathcal{D}$

$k++$

<— Which task $\mathcal{T}_j$ to choose?

- randomly
- task(s) in which the trajectory gets high reward

Eysenbach et al. Rewriting History with Inverse RL
Li et al. Generalized Hindsight for RL

When can we apply relabeling?

- reward function form is known, evaluatable
- dynamics consistent across goals/tasks
- using an off-policy algorithm*

Kaelbling. Learning to Achieve Goals. IJCAI '93
Andrychowicz et al. Hindsight Experience Replay. NeurIPS '17

Hindsight relabeling for goal-conditioned RL

Example: goal-conditioned RL, simulated robot manipulation

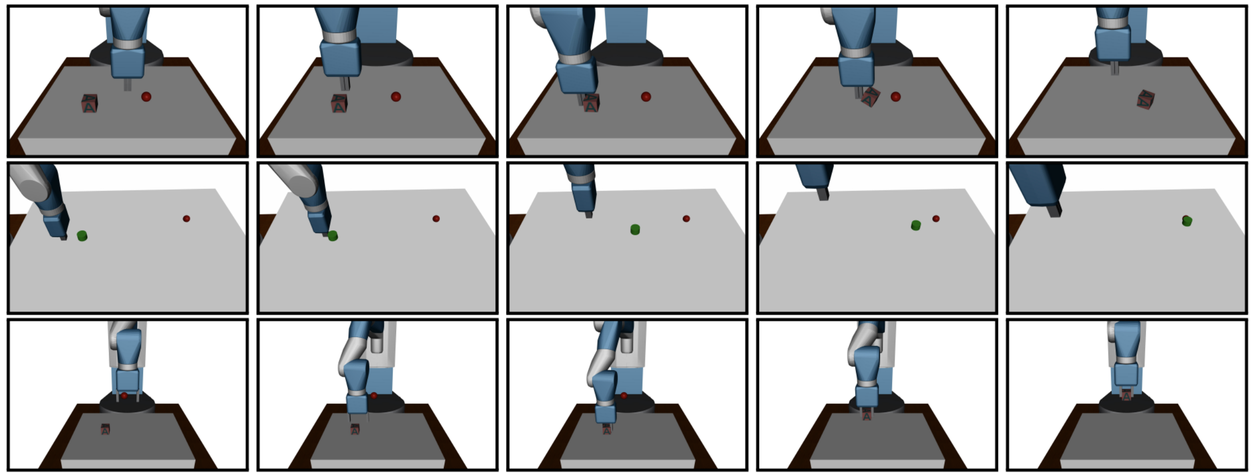
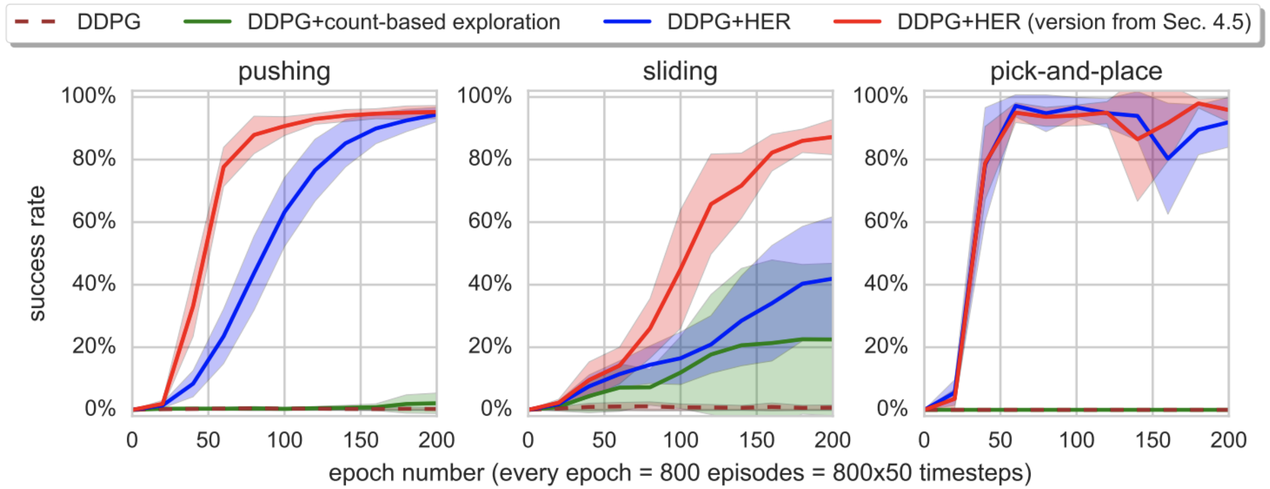


Figure 2: Different tasks: *pushing* (top row), *sliding* (middle row) and *pick-and-place* (bottom row). The red ball denotes the goal position.



Kaelbling. Learning to Achieve Goals. IJCAI '93

Andrychowicz et al. Hindsight Experience Replay. NeurIPS '17

Time Permitting: What about image observations?

Recall: need a distance function between current and goal state!

$$r'_t = -d(\mathbf{s}_t, \mathbf{s}_T)$$

Use binary 0/1 reward? Sparse, but accurate.

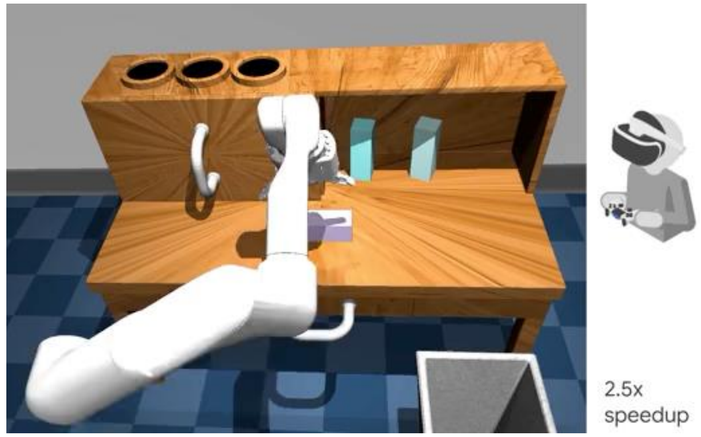
Random, **unlabeled interaction** is ***optimal*** under the 0/1 reward of reaching the last state.

Can we use this insight for **better learning**?

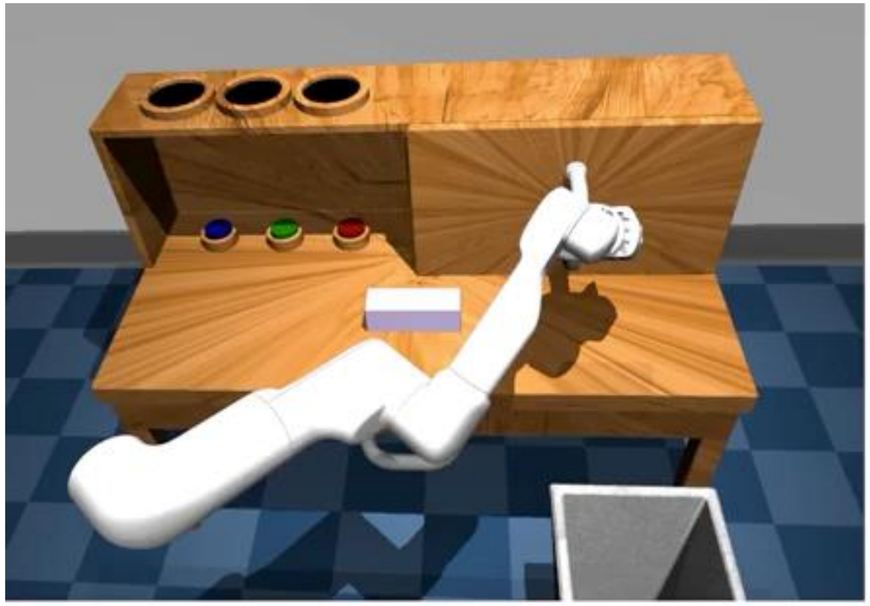
If the data is **optimal**, can we use **supervised imitation learning**?

1. Collect data $\mathcal{D}_k = \{(\mathbf{s}_{1:T}, \mathbf{a}_{1:T})\}$ using some policy
2. Perform hindsight relabeling:
   a. Relabel experience in $\mathcal{D}_k$ using last state as goal:
   $\mathcal{D}'_k = \{(\mathbf{s}_{1:T}, \mathbf{a}_{1:T}, \mathbf{s}_T, r'_{1:T}\}$ where $r'_t = -d(\mathbf{s}_t, \mathbf{s}_T)$
   b. Store relabeled data in replay buffer $\mathcal{D} \leftarrow \mathcal{D} \cup \mathcal{D}'_k$
3. Update policy using **supervised imitation on** replay buffer $\mathcal{D}$

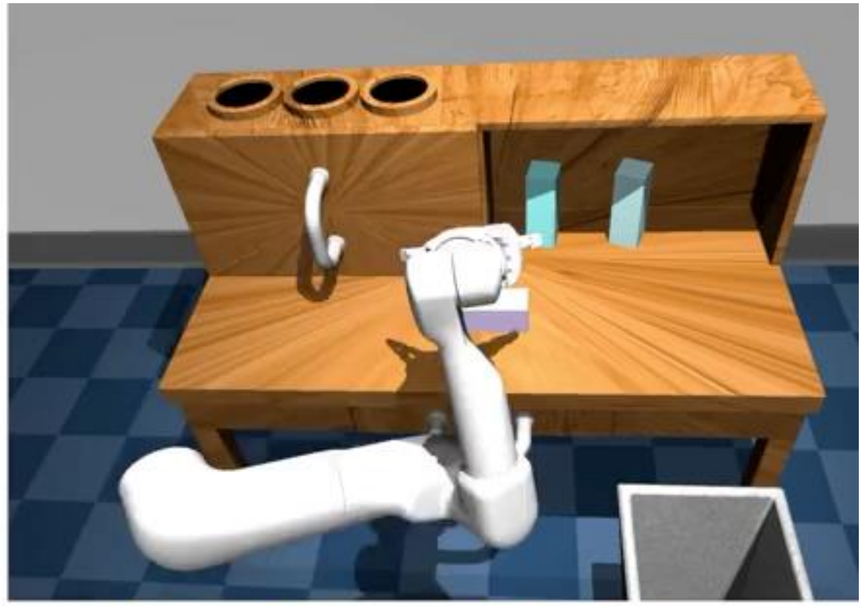
Collect data from "human play", perform goal-conditioned imitation.



2.5x
speedup



Goal



Single Play-LMP policy

Lynch, Khansari, Xiao, Kumar, Tompson, Levine, Sermanet. Learning Latent Plans from Play. '19

The Plan

Multi-task reinforcement learning problem

Policy gradients & their multi-task/meta counterparts

Q-learning

Multi-task Q-learning

How **data** can be **shared** across tasks.

Many Remaining Questions: The Next Three Weeks

How can we use a **model** in multi-task RL?	Model-based RL - **Oct 19**
What about **meta-RL** algorithms?	Meta-RL - **Oct 21**
Can we learn **exploration strategies** across tasks?	Meta-RL: Learning to explore - **Oct 26**
What about **hierarchies** of tasks?	Hierarchical RL - **Nov 2**

Additional RL Resources

Stanford CS234: Reinforcement Learning

UCL Course from David Silver: Reinforcement Learning

Berkeley CS285: Deep Reinforcement Learning